Project one was a challenge as I've never dived so deep into complex coding practices with C. Waitpid was a challenge I experienced as I had never used the function. Understanding the status of the waitpid function was challenging at first. I could not figure out the logic behind printing out the status code of the child process when it finished. It took TAs help to figure it out. Execvp was another difficulty I faced in this project. Specifically I wasn't familiar with pointers and the difference between char/string arrays in C. I tried for the longest time to put a char array into the second argument of execvp. To no avail obviously. I spent a night researching the difference as well as pointers in C. The next day was when I realized the problem and one of the TAs guided me towards a solution in my code.

My implementation of this project starts in the main function. I have a for loop for changing the header of the bash shell. After that we drop into the main while loop where all of the functionality is located. I scan for input with fgets and store that in a char array. Next, we go into a switch statement that is cased with an integer for each possible input in the shell. The switch statement has a function call at the top that parses that char array we got from fgets. The function called switchparser. takes the char array and tokenizes it using strtok. Then a series of else if statements determine an integer value based off of the command inputted by the user. The function then returns that integer value into the switch statement for command execution. All of the commands are contained within the switch statement except for execvp which has its own function. Within switchparser execvp has a few extra steps. I copy data from the tokenized input to two different arrays. The first is a simple char array and the second is a 2D char array. These are global variables that are used in the execvp function called execute. In the execute function I first dynamically allocate a char array for the specific size of the number of arguments in the user input plus a NULL character. Once that's done I have an if statement that forks into a child process where the command and other pertinent information is printed to the screen. Such as the PID of the child. Once the child is finished the parent can resume (waitpid). Which is when the PID and exit status code are printed onto the screen. After that the program starts a new iteration of the while loop in main.